


Introduction


Altera provides various internal memory (RAM and ROM) features to address the memory requirements of today's system-on-a-programmable-chip (SOPC) designs.

You can use the following methods to create the memory with the features you desire:

- Quartus® II MegaWizard™ Plug-In Manager
- Memory inferring from HDL code
- Manual instantiation of memory megafunctions

Altera recommends you to use MegaWizard Plug-In Manager to create internal memory compared to other methods.

 For general information about the Quartus II MegaWizard Plug-In Manager, refer to the *Megafunction Overview User Guide*.

 For more information about memory inferring from HDL code, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

The following sections describe the various memory features that you can configure through the MegaWizard interface:

- “Memory Modes” on page 2
- “Memory Block Types” on page 5
- “Write and Read Operations Triggering” on page 6
- “Port Width Configuration (Memory Depth × Data Width)” on page 8
- “Mixed-width Port” on page 9
- “Maximum Block Depth” on page 9
- “Clocking Modes and Clock Enable” on page 11
- “Address Clock Enable” on page 12
- “Byte Enable” on page 13
- “Asynchronous Clear” on page 14
- “Read Enable” on page 15
- “Read-During-Write” on page 16
- “Power-Up Conditions and Memory Initialization” on page 17
- “Error Correction Code (ECC)” on page 18
- “Design Example: External ECC Implementation with True-Dual-Port RAM” on page 19
- “Ports and Parameters” on page 28

Memory Modes

Altera provides the following memory modes that you can create using the corresponding MegaWizard interfaces (listed in brackets):

- Single-port RAM (RAM:1-Port)
- Simple dual-port RAM (RAM: 2-Port)
- True dual-port RAM (RAM:2-Port)
- Tri-port RAM (RAM:3-Port)
- Single-port ROM (ROM:1-Port)
- Dual-port ROM (ROM:2-Port)

You can find these MegaWizard interfaces under the Memory Compiler category when you launch the MegaWizard Plug-In Manager.

In general, the memory block contains two address port (port A and port B) with their respective output data ports, and you can use them for read and write operations depending on your memory modes. This section shows the different memory modes with their input and output ports in a block diagram.

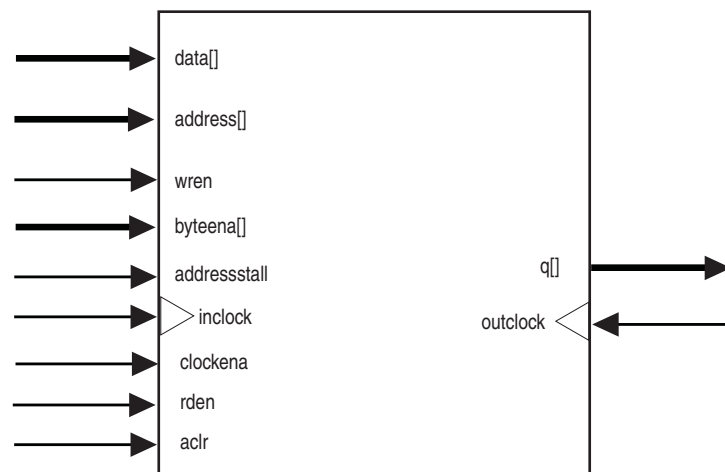


The input and output ports shown in the block diagrams are referring to the ports of the wrapper that contains the memory megafunction instantiated in it. The port name reflects the usage related to the memory features you created. For example, `byteena` relates to the byte enable feature, `addressstall` relates to the address clock enable features, and so on. You can find more details about the various memory features in the following sections.

In a single-port RAM, the read and write operations share the same address at port A, and the data is read from output port A.

[Figure 1](#) shows a block diagram of a typical single-port RAM.

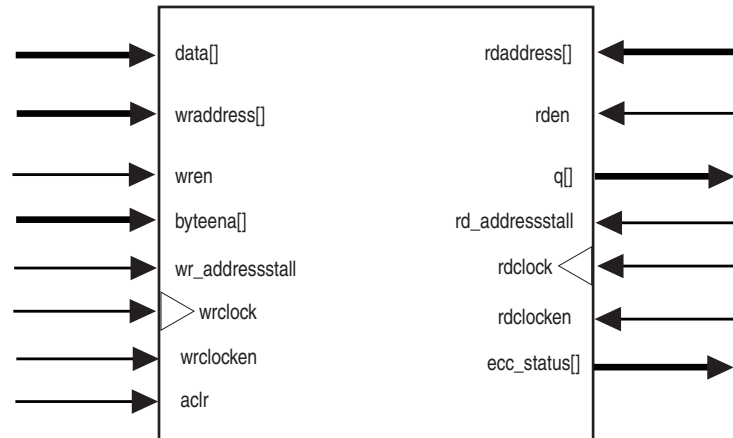
Figure 1. Single-Port RAM



In simple dual-port RAM mode, a dedicated address port is available for each read and write operation (one read port and one write port). A write operation uses write address from port A while read operation uses read address and output from port B.

Figure 2 shows the block diagram of a simple dual-port RAM.

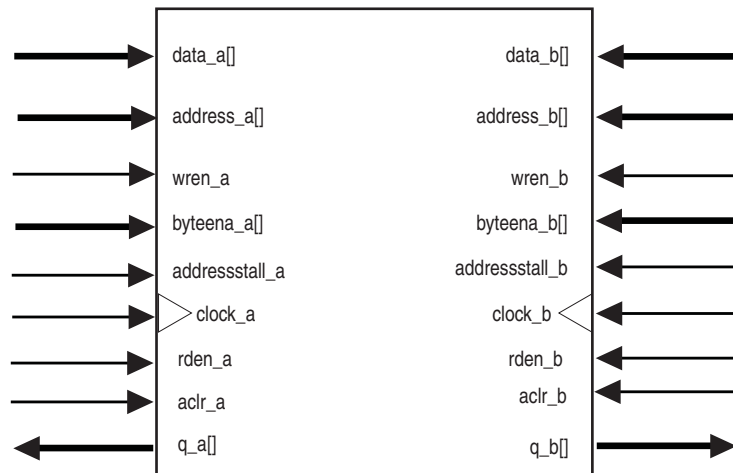
Figure 2. Simple Dual-Port RAM



In true dual-port RAM mode, two address ports are available for read or write operation (two read/write ports). In this mode, you can write to or read from the address of port A or port B, and the data read is shown at the output port with respect to the read address port.

Figure 3 shows the block diagrams of a true dual-port RAM.

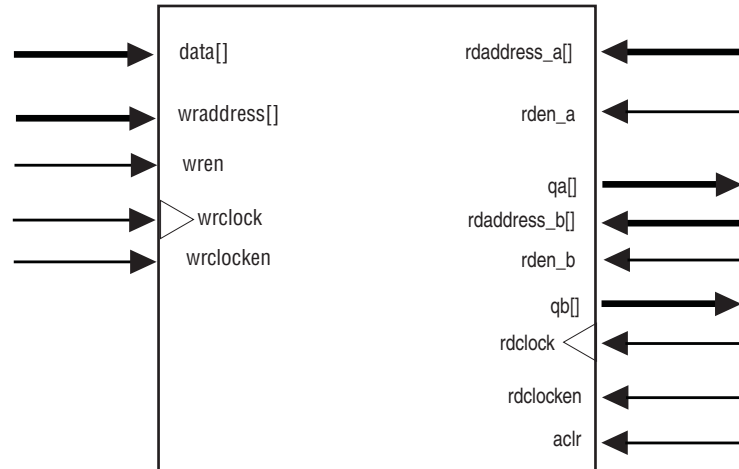
Figure 3. True Dual-Port RAM



In a tri-port RAM, two read address ports and one write address port are available (two read ports and one write port).

Figure 4 shows the block diagrams of a tri-port RAM.

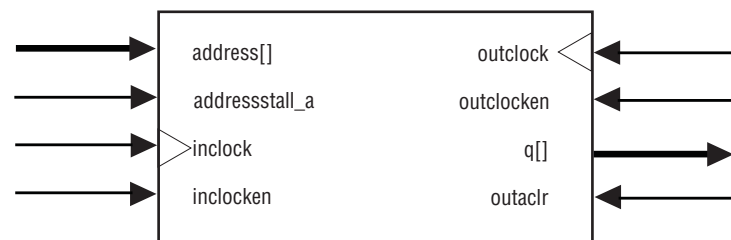
Figure 4. Tri-Port RAM



In single-port ROM, only one address port is available for read operation.

Figure 5 shows the block diagrams of a typical single-port ROM. The dual-port ROM has almost similar functional ports as single-port ROM. The difference is dual-port ROM has an additional address port for read operation.

Figure 5. Single-Port ROM



Memory Block Types

The embedded memory blocks in Altera devices feature the TriMatrix memory structure that provides three different sizes of embedded SRAM. Different device families support different sizes of the TriMatrix embedded memory blocks. [Table 1](#) shows the type of TriMatrix memory blocks in various device families.

Table 1. TriMatrix Embedded Memory Blocks in Altera Devices

Device Family	Types of TriMatrix Memory Blocks
Stratix [®] , Stratix GX, Stratix II, Stratix II GX, Cyclone [®] , Cyclone II (1), HardCopy [®] II (2), and Arria [®] GX	M512 blocks (512 bits) (5) M4K blocks (4 Kbits) M-RAM blocks (512 Kbits) (6)
Stratix III, HardCopy III, Cyclone III (3), Arria II GX (4), and newer devices	MLAB blocks (640 bits) (7)(8) M9K blocks (9 Kbits) M144K blocks (144 Kbits)

Notes to Table 1:

- (1) Cyclone and Cyclone II devices do not have M512 blocks and M-RAM blocks.
- (2) HardCopy II devices do not have M512 blocks.
- (3) Cyclone III devices do not have MLAB blocks and M144K blocks.
- (4) Arria II GX devices do not have M144K blocks.
- (5) M512 blocks are not supported in true dual-port RAM mode, and dual-port ROM mode.
- (6) M-RAM blocks are not supported in ROM mode.
- (7) For Stratix III devices, MLAB blocks are 320-bit in RAM mode and 640-bit in ROM mode.
- (8) MLAB blocks are not supported in simple dual-port RAM mode with mixed-width port feature, true dual-port RAM mode, and dual-port ROM mode.



For more information about TriMatrix memory blocks and the specifications, refer to the *TriMatrix Embedded Memory Blocks* chapter in your target device handbook.

From the MegaWizard interface, you can implement your memory in any of the supported TriMatrix memory blocks available based on your target device. You can also choose to implement the memory using logic cells, or allow the software to automatically select the appropriate TriMatrix memory resource.

If you want to specifically select the TriMatrix memory block, obtain more information about the features of your selected TriMatrix memory block in your target device, such as the maximum performance, supported configurations (depth × width), byte enable, power-up condition, and the write and read operation triggering. As compared to TriMatrix memory resources, using logic cells to create memory reduces the design performance and utilizes more area. This implementation is normally used when you have used up all the TriMatrix memory resources. When logic cells are used, the MegaWizard provides you with the following two types of logic cell implementations:

- Default logic cell style—the write operation triggers (internally) on the rising edge of the write clock and have continuous read. This implementation uses less logic cells and is faster, but it is not fully compatible with the Stratix M512 emulation style.
- Stratix M512 emulation logic cell style—the write operation triggers (internally) on the falling edge of the write clock and performs read only on the rising edge of the read clock.

To obtain proper implementation based on the memory configuration you set, allow the Quartus II software to automatically choose the memory type. This gives the compiler the flexibility to place the memory function in any available memory resources based on the functionality and size.



To identify the type of memory block that the software selects to create your memory, refer to the fitter report after compilation.

When you set the memory block type to **Auto**, the compiler favors larger block types that can support the memory capacity you require in a single TriMatrix memory block. This setting gives the best performance and requires no logic elements (LEs) for glue logic. When you create the memory with specific TriMatrix memory blocks, such as M9K, the compiler is still able to emulate wider and deeper memories than the block type supported natively. The compiler spans multiple TriMatrix memory blocks (only of the same type) with glue logic added in the LEs as needed.

Write and Read Operations Triggering

The TriMatrix memory blocks vary slightly in its supported features and behaviors. One important variation is the difference in the write and read operations triggering for different types of TriMatrix memory blocks.

Table 2 shows the write and read operations triggering for different TriMatrix memory blocks.

Table 2. Write and Read Operations Triggering for TriMatrix Memory Blocks

TriMatrix Memory Blocks	Write Operation (1)	Read Operation
M144K	Rising clock edges	Rising clock edges
M9K	Rising clock edges	Rising clock edges
MLAB	Falling clock edges	Rising clock edges (2)
M-RAM	Rising clock edges	Rising clock edges
M4K	Falling clock edges	Rising clock edges
M512	Falling clock edges	Rising clock edges

Notes to Table 1:

- (1) Write operation triggering is not applicable to ROMs.
- (2) MLAB supports continuous reads. For example, when you write a data at the write clock rising edge and after the write operation is complete, you see the written data at the output port without the need for a read clock rising edge.

It is important that you understand the write operation triggering to avoid potential write contentions that can result in unknown data storage at that location.

Figure 6 and Figure 7 show the valid write operation that triggers at the rising and falling clock edge, respectively.

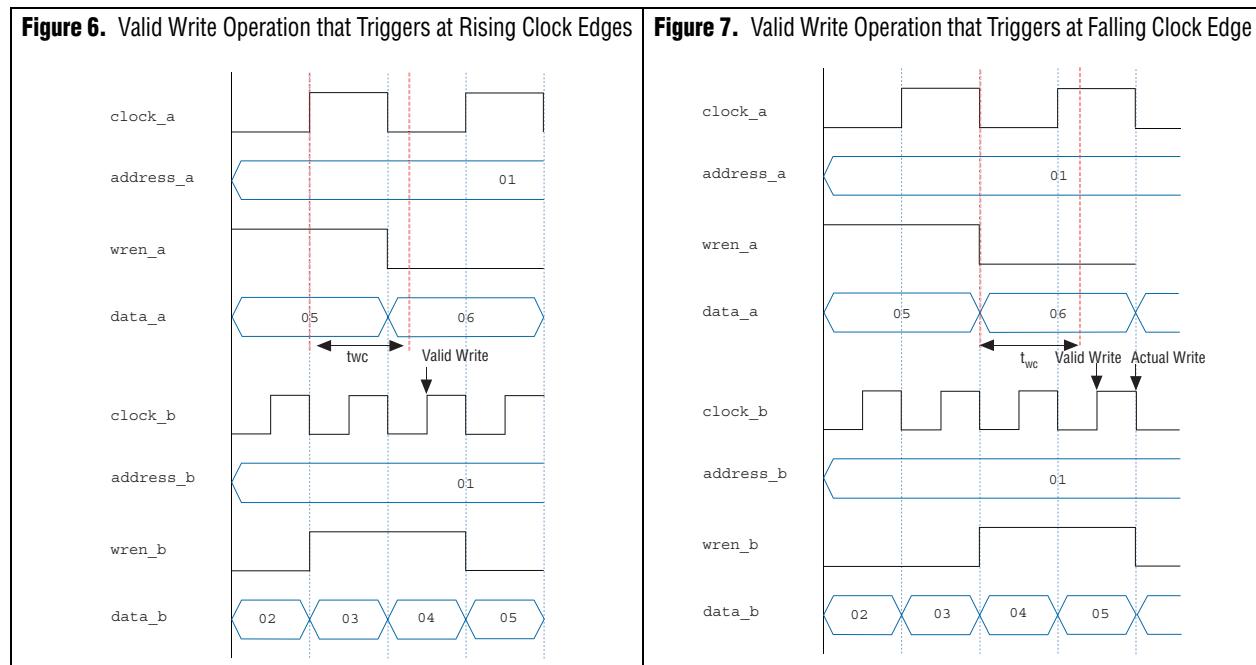


Figure 6 assumes that t_{wc} is the maximum write cycle time interval. Write operation of data 03 through port B does not meet the criteria and causes write contention with the write operation at port A, which result in unknown data at address 01. The write operation at the next rising edge is valid because it meets the criteria and data 04 replaces the unknown data.

Figure 7 assumes that t_{wc} is the maximum write cycle time interval. Write operation of data 04 through port B does not meet the criteria and therefore causes write contention with the write operation at port A that result in unknown data at address 01. The next data (05) is latched at the next rising clock edge that meets the criteria and is written into the memory block at the falling clock edge.



Data and addresses are latched at the rising edge of the write clock regardless of the different write operation triggering.

Port Width Configuration (Memory Depth × Data Width)

The port width configuration is defined as the memory depth (number of words) × the width of the data input bus.


Table 3 shows the supported port width configuration (per memory block) for TriMatrix memory blocks in Stratix III devices.

Table 3. Port Width Configuration for TriMatrix Memory Blocks in Stratix III Devices

MLABs	M9K	M144K
16 × 8	8 K × 1	16 K × 8
16 × 9	4 K × 2	16 K × 9
16 × 10	2 K × 4	8 K × 16
16 × 16	1 K × 8	8 K × 18
16 × 18	1 K × 9	4 K × 32
16 × 20	512 × 16	4 K × 36
—	512 × 18	2 K × 64 (1)
—	256 × 32 (1)	2 K × 72 (1)
—	256 × 36 (1)	—

Note to Table 3:


(1) Only applicable for single-port RAM, simple-dual port RAM, and single-port ROM.

 For more information about the supported port width configuration for different TriMatrix memory blocks, refer to the *TriMatrix Embedded Memory Blocks* chapter in your target device handbook.

If your port width configuration (either the depth or the width) is more than the amount a TriMatrix memory block can support, additional memory blocks (of the same type) are used. For example, if you configure your M9K as 512 × 36, which exceeds the supported port width of 512 × 18, two M9Ks are used to implement your RAM.

In addition to the supported configuration provided, you can set the memory depth to a non-power of two, but the actual memory depth allocated can vary. The variation depends on the type of resource implemented.

If the memory is implemented in TriMatrix memory blocks, setting a non-power of two for the memory depth reflects the actual memory depth. If the memory is implemented in logic cells (and not using Stratix M512 emulation logic cell style that can be set through the MegaWizard interfaces), setting a non-power of two for the memory depth does not reflect the actual memory depth. In this case, you write to or read from up to $2^{\text{address_width}}$ memory locations even though the memory depth you set is less than $2^{\text{address_width}}$. For example, if you set the memory depth to 3, and the RAM is implemented using logic cells, your actual memory depth is 4.

 When you implement your memory using TriMatrix memory blocks, you can check the actual memory depth by referring to the fitter report.

Mixed-width Port

Only dual-port RAM and dual-port ROM support mixed-width port configuration for all memory block types except when they are implemented with LEs. The support for mixed-width port depends on the width ratio between port A and port B. In addition, the supporting ratio varies for various memory modes, memory blocks, and target devices.



MLABs do not have native support for mixed width operation, thus the option to select MLABs is disabled in the MegaWizard interface. However, the Quartus II software can implement mixed width memories in MLABs by using more than one MLAB. Therefore, if you select **AUTO** for your memory block type, it is still possible to implement mixed-width port memory using multiple MLABs.



For more information about width ratio that supports mixed-width port, refer to your relevant device handbook.

Memory depth of 1 word is not supported in simple dual-port and true dual-port RAMs with mixed-width port. The RAM MegaWizard interface prompts an error message when the memory depth is less than 2 words. For example, if the width for port A is 4 bits and the width for port B is 8 bits, the smallest depth supported by the RAM is 4 words. This configuration results in memory size of 16 bits (4x4) and can be represented by memory depth of 2 words for port B. If you set the memory depth to 2 words that results in memory size of 8 bits (2x4), it can only be represented by memory depth of 1 word for port B, and therefore the width of the port is not supported.

Maximum Block Depth

You can limit the maximum block depth of the TriMatrix memory block you use. The memory block can be sliced to your desired maximum block depth. For example, the capacity of an M9K block is 9,216 bits, and the default memory depth is 8K, in which each address is capable of storing 1 bit (8K × 1). If you set the maximum block depth to 512, the M9K block is sliced to a depth of 512 and each address is capable of storing up to 18 bits (512 × 18).

You can use this option to save power usage in your devices. However, this parameter might increase the number of LEs and affects the design performance.

[Table 4](#) shows the estimated dynamic power usage for different slice type that is applied to an 8K × 36 (M9K RAM block) design in a Stratix III EP3SE50 device.

Table 4. Power Usage Setting for 8K × 36 (M9K) Design of a Stratix III Device

M9K Slice Type	Dynamic Power (mW)	ALUT Usage	M9Ks
8K × 1 (default setting)	51.49	0	36
4K × 2	20.28 (39%)	38	36
2K × 4	10.80 (21%)	44	36
1K × 9	6.08 (12%)	125	32
512 × 18	4.51 (9%)	212	32
256 × 36	6.36 (12%)	467	32

When the RAM is sliced shallower, the dynamic power usage decreases. However, for a RAM block with a depth of 256, the power used by the extra LEs starts to outweigh the power gain achieved by shallower slices.

You can also use this option to reduce the total number of memory blocks used (but at the expense of LEs). From [Table 4](#), the 8K × 36 RAM uses 36 M9K RAM blocks with a default slicing of 8K × 1. By setting the maximum block depth to 1K, the 8K × 36 RAM can fit into 32 M9K blocks.

The maximum block depth must be in a power of two, and the valid values vary among different TriMatrix memory blocks.

[Table 5](#) shows the valid range of maximum block depth for different TriMatrix memory blocks.

Table 5. Valid Range of Maximum Block Depth for Different TriMatrix Memory Blocks

TriMatrix Memory Blocks	Valid Range (1)
M144K	4,096 – 13,1072
M9K	128 – 8,192
MLAB	32 – 64 (2)
M512	32 – 64
M4K	128 – 4,096
M-RAM	4,096 – 65,536

Notes to Table 5:

- (1) The maximum block depth must be in a power of two.
- (2) The maximum block depth setting for MLAB is not available for Stratix III devices.

The MegaWizard interface prompts an error message if you enter an invalid value for the maximum block depth. You are advised to set the value to **Auto** if you are not sure of the appropriate maximum block depth to set or the setting is not important for your design. This setting enables the compiler to select the maximum block depth with the appropriate port width configuration for the type of TriMatrix memory block of your memory.


Clocking Modes and Clock Enable

Altera memory supports various types of clocking modes depending on the memory mode you select.

Table 6 shows the supported types of clocking modes in most Altera devices.

Table 6. Supported Types of Clocking Modes in Most Altera Devices

Clocking Modes	Single-port RAM	Simple Dual-port RAM	True Dual-port RAM	Tri-port RAM	Single-port ROM	Dual-port ROM
Single clock	✓	✓	✓	✓	✓	✓
Read/Write	X	✓	X	✓	X	X
Input/Output	✓	✓	✓	✓	✓	✓
Independent	X	X	✓	X	X	✓


 Asynchronous clock mode is only supported in MAX series of devices, and not supported by Stratix and newer devices. However, Stratix III and newer devices support asynchronous read memory for simple dual-port RAM mode if you choose MLAB memory block with unregistered rdaddress port.

In the single clock mode, a single clock can be used together with a clock enable to control all registered ports or selected registered ports of the memory blocks.

In the read/write clock mode, a separate clock is available for each read and write port. The read clock controls all the registered read ports (data output, read address, and read-enable ports) and the write clock controls all the registered write ports (data input, write address, write enable, and byte enable ports).

In input/output clock mode, a separate clock is available for each input and output port. The input clock controls all registered input ports (data input, addresses, byte enables, read enables, and write-enables ports) to the memory and the output clock controls the output registered ports (data output).

In the independent clock mode, a separate clock is available for each port (A and B). Clock A controls all registered ports of port A, while clock B controls all registered ports of port B.

 You can create independent clock enable for different input and output registers to control the shut down of a particular register for power saving purposes. From the MegaWizard interface, click **More Options** (beside the clock enable option) to set the available independent clock enable that you prefer.

Address Clock Enable

The address clock enable (`addressstall`) port is an active high asynchronous control signal used to hold the previous address value for as long as the signal is enabled. For dual-port RAMs and dual-port ROMs, you can create independent address clock enable for each address port.

Figure 8 and Figure 9 show the results of address clock enable signal during the read and write operations, respectively.

Figure 8. Address Clock Enable During Read Operation

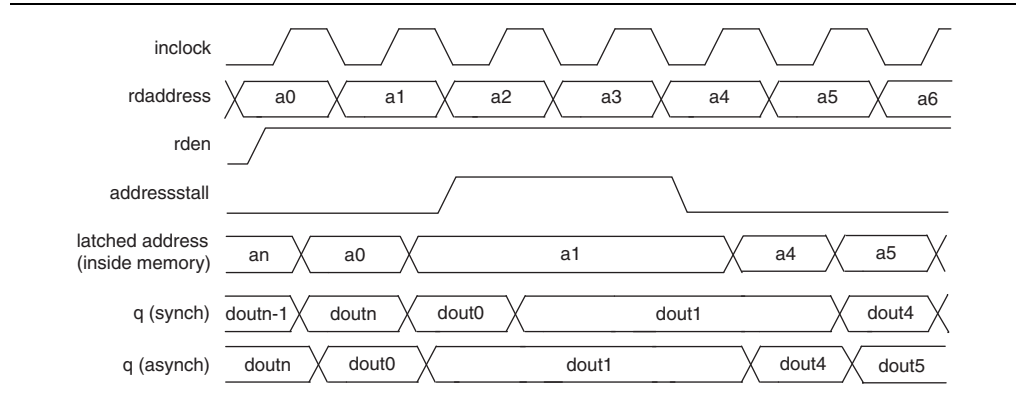
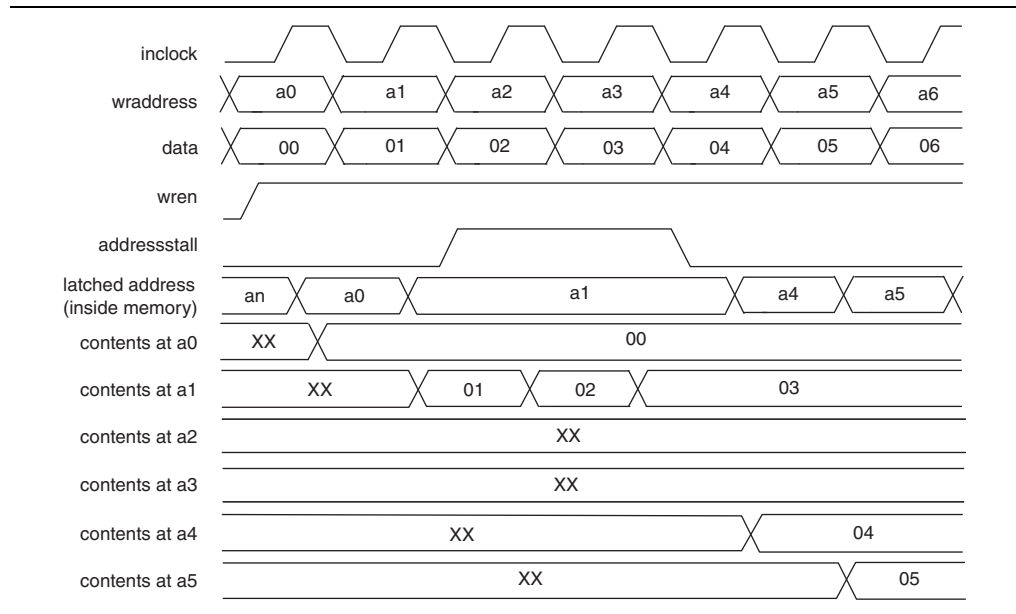




Figure 9. Address Clock Enable During Write Operation



 To configure the address clock enable feature, click **More Options** located beside the clock enable option on the RAM MegaWizard interface. Turn on the option to create the `addressstall` port to enable the feature.

 The address clock enable feature is only supported by Stratix II and newer devices, and for all memory modes excluding tri-port RAM.

Byte Enable

All TriMatrix memory blocks that are implemented as RAMs, support byte enables that mask the input data so that only specific bytes, nibbles, or bits of data are written. The unwritten bytes or bits retain the previously written value.

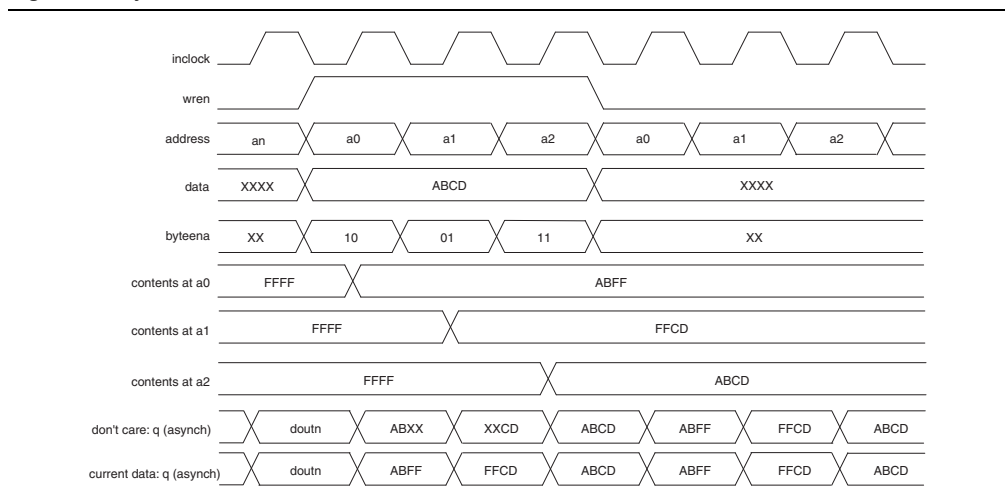
Byte enable port operates in a one-hot fashion, with the least significant bit (LSB) of the byte-enable port corresponding to the least significant byte of the data bus. For example, if you use a RAM block in x18 mode and the byte-enable port is 01, data [8..0] is enabled and data [17..9] is disabled. Similarly, if the byte-enable port is 11, both data bytes are enabled.

You can specifically define and set the size of a byte for the byte-enable port. The valid values are 5, 8, 9, and 10, depending on the type of TriMatrix memory blocks. The values of 5 and 10 are only supported by MLAB.

To create a byte-enable port, the width of the data input port must be a multiple of the size of a byte for the byte-enable port. For example, if you use an MLAB memory block, the byte enable is only supported if your data bits are multiples of 5, 8, 9 or 10, that is 10, 15, 16, 18, 20, 24, 25, 27, 30, and so on. If the width of the data input port is 10, you can only define the size of a byte as 5. In this case, you get a 2-bit byte-enable port, each bit controls 5 bits of data input written. If the width of the data input port is 20, then you can define the size of a byte as either 5 or 10. If you define 5 bits of input data as a byte, you get a 4-bit byte-enable port, each bit controls 5 bits of data input written. If you define 10 bits of input data as a byte, you get a 2-bit byte-enable port, each bit controls 10 bits of data input written.

Figure 10 shows the results of the byte enable on the data that is written into the memory, and the data that is read from the memory.

Figure 10. Byte Enable Functional Waveform



When a byte-enable bit is deasserted during a write cycle, the corresponding masked byte of the q output can appear as a “Don't Care” value or the current data at that location. This selection is only available if you set the read-during-write output behavior to **New Data**.



For more information about the masked byte and the q output, refer to “Read-During-Write” on page 16.

Asynchronous Clear

Table 7 shows the asynchronous clear effects on the input ports for different devices in different memory settings.

Table 7. Asynchronous Clear Effects on the Input Ports for Different Devices in Different Memory Settings

Memory Mode	Stratix, Stratix GX, and Cyclone	Stratix II, Stratix II GX, HardCopy II, Cyclone II, and Arria GX	Stratix III, HardCopy III, Cyclone III, Arria II GX, and newer devices
Single-port RAM	All registered input ports can be affected except for the following ports and conditions: <ul style="list-style-type: none"> wren port for M512 data/wren/address ports for MRAM (byteena port can be affected) LCs are implemented (1) 	All registered input ports are not affected (1)	All registered input ports are not affected (1)
Single dual-port RAM and True dual-port RAM	All input registered ports can be affected except for MRAM	All registered input ports are not affected	Only registered input read address port can be affected.
Tri-port RAM	All registered input ports can be affected (2)	All registered input ports are not affected	Only registered input read address port can be affected (excluding M144K).
Single-port ROM	Registered address input port can be affected	All registered input ports are not affected	Registered input address port can be affected
Dual-port ROM	Registered address input port can be affected	All registered input ports are not affected	All registered input ports are not affected

Notes to Table 7:

- (1) When LCs are implemented in this memory mode, registered output port is not affected.
(2) For MRAM, only the read address input ports can be affected



During a read operation, clearing the input read address asynchronously corrupts the memory contents. The same effect applies to a write operation if the write address is cleared.

You can select the ports that are affected by the asynchronous clear signal using the MegaWizard interface. Click **More Options** located next to the **asynchronous clear** option, and the asynchronous clear page appears. The page shows you the ports that are disabled. These disabled ports are not affected by the asynchronous clear signal. On the same page, you can turn on the available ports that can be affected by the asynchronous clear signal.

The TriMatrix memory blocks in the Stratix III, Cyclone III, HardCopy III, Arria II GX, and newer device families support the asynchronous clear feature used on the output latches and output registers. The asynchronous clear feature allows you to clear the outputs even if the *q* output port is not registered. This feature, however, is not supported in MLAB memory blocks.

Read Enable

Support for the read enable feature depends on the target device, memory block type, and the memory mode you select. [Table 8](#) shows the memory configurations for the different device families that support the read enable feature.

Table 8. Read-Enable Support in Different Device Families

Memory Modes	Stratix III, HardCopy III, Cyclone III, Arria II GX, and newer Devices		Other Stratix and Cyclone Devices	
	M9K, M144K	MLAB	M512, M4K	M-RAM
Single-port RAM	✓	X	X	X
Simple dual-port RAM	✓	X	✓	X
True dual-port RAM	✓	X	X	X
Tri-port RAM	✓	X	✓	X
Single-port ROM	✓	X	X	X
Dual-port ROM	X	X	X	X

If you create the read-enable port and perform a write operation (with the read enable port deasserted), the data output port retains the previous values that are held during the most recent active read enable. If you activate the read enable during a write operation, or if you do not create a read-enable signal, the output port shows the new data being written, the old data at that address, or a “Don't Care” value when read-during-write occurs at the same address location. You can set the output behavior from the MegaWizard interface.



For more information about the read-during-write output behavior, refer to the [“Read-During-Write”](#) on page 16.

Read-During-Write

The read-during-write (RDW) occurs when a read and a write target the same memory location at the same time. You can use the RAM MegaWizard interface to configure the RDW output behavior of your RAM. There are two types of RDW operations available—same-port and mixed-port.

The same-port RDW occurs when the input and output of the same port access the same address location with the same clock. The mixed-port RDW occurs when one port reads and another port writes to the same address location with the same clock. The available output choices for the RDW behavior vary, depending on the types of RDW and TriMatrix memory block in use.

Table 9 shows the available output choices for the same-port, and mixed-port RDW for different TriMatrix memory blocks.


Table 9. Output Choices for the Same-Port and Mixed-Port Read-During-Write

Memory Block Types	Single-port RAM (1)	Simple dual-port RAM (2)	True dual-port RAM	
	Same port RDW	Mixed-port RDW	Same port RDW (3)	Mixed-port RDW (4)
M512	No MW (5)	Old Data	NA	
M4K		Don't Care	No MW (5)	Old Data
M-RAM		Don't Care		Don't Care
MLAB	Don't Care	Old Data Don't Care	NA MLAB is not supported in tdp RAM	
M9K	Don't Care	Old Data	New Data (6)	Old Data
M144K	New Data (6) Old Data	Don't Care	Old Data	Don't Care
LCs	No MW (5)		NA	

Notes to Table 9:

- (1) Single-port RAM only supports same-port RDW, and the clocking mode must be either single clock mode, or input/output clock mode.
- (2) Simple dual-port RAM only supports mixed-port RDW, and the clocking mode must be either single clock mode, or input/output clock mode.
- (3) The clocking mode must be either single clock mode, input/output clock mode, or independent clock mode.
- (4) The clocking mode must be either single clock mode, or input/output clock mode.
- (5) There is no option page available from the MegaWizard interface in this mode. By default, the new data flows through to the output.
- (6) There are two types of new data behavior for same-port RDW that you can choose from the MegaWizard interface. When byte enable is applied, you can choose to read old data, or 'X' on the masked byte. The respective parameter values are:
 - **NEW_DATA_WITH_NBE_READ** for old data on masked byte
 - **NEW_DATA_NO_NBE_READ** for x on masked byte.

 The RDW old data mode is not supported when the Error Correction Code (ECC) is engaged or when you configure your memory as tri-port RAM.

 If you are not concerned about the output when RDW occurs, you can select **Don't Care**. Selecting **Don't Care** increases the flexibility in the type of memory block being used, provided you do not assign block type when instantiating the memory block. You also get a potential performance gain by selecting **Don't Care**.

Power-Up Conditions and Memory Initialization

Power-up conditions depend on the type of TriMatrix memory blocks in use and whether or not the output port is registered.

Table 10 shows the power-up conditions in the different types of TriMatrix memory blocks.

Table 10. Power-Up Conditions for Different TriMatrix Memory Blocks

TriMatrix Memory Blocks	Power-Up Conditions
M512	Outputs cleared
M4K	Outputs cleared
M-RAM	Outputs cleared if registered, otherwise unknown
MLAB	Outputs cleared if registered, otherwise reads memory contents
M9K	Outputs cleared
M144K	Outputs cleared

The outputs of M512, M4K, M9K, and M144K blocks always power-up to zero, whether the output registers are used or bypassed. Even if a memory initialization file is used to pre-load the contents of the memory block, the output is still cleared.

MLAB and M-RAM blocks power-up to zero only if output registers are used. If output registers are not used, MLAB blocks power-up to read the memory contents while M-RAM blocks power-up to an unknown state.



When the memory block type is set to **Auto** in the MegaWizard interface, the compiler is free to choose any memory block type, in which the power-up value depends on the chosen memory block type. To identify the type of memory block the software selected to implement your memory, refer to the fitter report after compilation.

All memory blocks (excluding M-RAM) support memory initialization via the Memory Initialization File (.mif) or Hexadecimal (Intel-format) file (.hex). You can include the files using the MegaWizard interface when you configure and build your RAM. For RAM, beside using the .mif file or the .hex file, you can initialize the memory to zero or 'X'. To initialize the memory to zero, select **No, leave it blank**. To initialize the content to 'X', turn on **Initialize memory content data to XX..X on power-up in simulation**. Turning on this option does not change the power-up behavior of the RAM but initializes the content to 'X'. For example, if your target memory block is M4K, the output is cleared during power-up (based on Table 10 on page 17). The content that is initialized to 'X' is shown only when you perform the read operation.

Error Correction Code (ECC)

The ECC features single-error-correction double-error detection (SECDED) implementation, in which it can detect and fix a single-bit error or detect two-bit errors (without fixing). The Stratix III and Stratix IV M144K memory blocks have built-in support for the ECC up to a data width of x64 for the simple dual-port mode.

However, this feature is not supported if the:

- mixed-width port feature is used
- byte-enable feature is engaged



The Mixed-port RDW for old data mode is not supported when the ECC feature is engaged. The result for RDW is "don't care"

The M144K ECC status is communicated via a three-bit status flag `eccstatus[2..0]`.

Table 11 shows the truth table for the ECC status flags.

Table 11. Truth Table for ECC Status Flags

Status	Eccstatus[2..0]
No error	000
Single error and fixed	011
Double error and no fix	101
Illegal	001
	010
	100
	11X



You can also use the `ALTECC_ENCODER` and the `ALTECC_DECODER` megafunctions to implement the ECC external to your memory blocks. For more information, refer to the *Integer Arithmetic Megafunctions User Guide*.

Design Example: External ECC Implementation with True-Dual-Port RAM

The ECC features are only supported internally in simple dual-port RAM by Stratix III and Stratix IV devices when the M144K is implemented. Therefore, this design example describes how ECC features can be implemented in other RAM modes, regardless of the type of device memory block used. It also demonstrates the features of the same-port and the mixed-port read-during-write behaviors.

This design example uses a true dual-port RAM and illustrates how the ECC feature can be implemented external to the RAM. The ALTECC_ENCODER and ALTECC_DECODER megafunctions are required as the ALTECC_ENCODER megafunction encodes the data input before writing the data into the RAM, while the ALTECC_DECODER megafunction decodes the data output from the RAM before transferring the data out to other parts of the logic.

In this design example, the raw data width is 8 bits and is encoded by the ALTECC_ENCODER megafunction block to produce a 13-bit width data that is written into the true dual-port RAM when write-enable signal is asserted. Because the RAM mode has two dedicated write ports, another encoder is implemented for the other RAM input port.

Two ALTECC_DECODER megafunction blocks are also implemented at each of the data output ports of the RAM. When the read-enable signal is asserted, the encoded data is read from the RAM address and decoded by the ALTECC_DECODER megafunction blocks, respectively. The decoder shows the status of the data as no error detected, single-bit error detected and corrected, or fatal error (more than 1-bit error).

This example also includes a "corrupt zero bit" control signal at port A of the RAM. When the signal is asserted, it changes the state of the zero-bit (LSB) encoded data before it is written into the RAM. This signal is used to corrupt the zero-bit data storing through port A, and examines the effect of the ECC features.



This design example describes how ECC features can be implemented with the RAM for cases in which the ECC is not supported internally by the RAM. However, the design examples might not represent the optimized design or implementation.

Design Files

The design examples in this user guide utilize the MegaWizard Plug-In Manager in the Quartus II software and are available on the [Literature and Technical Documentation](#) page of the Altera website. The files are located under the following sections:

- On the [Quartus II Development Software Literature](#) page, expand the **Using Megafunctions** section and then expand the **Memory Compiler** section
- [Literature: User Guides](#) section

The following design files can be found in **Internal_Memory_DesignExample.zip**:

- **ecc_encoder.v**
- **ecc_decoder.v**
- **true_dp_ram.v**
- **top_dpram.v**
- **true_dp_ram.vt**
- **true_dp.do**
- **true_dp.qar** (Quartus II design file)

Configuration Settings

The **ecc_encoder.v** is a design variation file for the ALTECC_ENCODER megafunction that is pre-configured with the settings shown in [Table 12](#).

Table 12. ALTECC_ENCODER Megafunction Settings

MegaWizard Page	Available Options	Configured Settings
3	Currently selected device family:	Stratix III
	How do you want to configure this module?	Configure this module as an ECC encoder
	How wide should the data be?	8 bits
	Do you want to pipeline the functions?	Yes, I want an output latency of 1 clock cycle
	Create an 'aclr' asynchronous clear port	Not selected
	Create a 'clocken' clock enable clock	Not selected

The **ecc_decoder.v** is a design variation file for the ALTECC_DECODER megafunction that is pre-configured with the settings shown in [Table 13](#).

Table 13. ALTECC_DECODER Megafunction Settings

MegaWizard Page	Available Options	Configured Settings
3	Currently selected device family:	Stratix III
	How do you want to configure this module?	Configure this module as an ECC decoder
	How wide should the data be?	13 bits
	Do you want to pipeline the functions?	Yes, I want an output latency of 1 clock cycle
	Create an 'aclr' asynchronous clear port	Not selected
	Create a 'clocken' clock enable clock	Not selected



For more information about the options available from the ALTECC MegaWizard Plug-In Manager, refer to [Integer Arithmetic Megafunctions User Guide](#).

The `true_dp_ram.v` is a design variation file for the true dual-port RAM (instantiated through the ALTSYNCRAM megafunction) that is pre-configured with the settings shown in Table 14.

Table 14. RAM:2-PORT MegaWizard Plug-In Settings

MegaWizard Page	Available Options	Configured Settings
3	Currently selected device family:	Stratix III
	How will you be using the dual port ram?	With two read/write ports
	How do you want to specify the memory size?	As a number of words
4	How many 8-bit words of memory?	16
	Use different data widths on different ports	Not selected
	How wide should the 'q_a' output bus be?	13
	What should the memory block type be?	M9K
	Set the maximum block depth to	Auto
5	Which clocking method do you want to use?	Single clock
	Create 'rden_a' and 'rden_b' read enable signals	Not selected
	Byte Enable Ports	Not selected
6	Which ports should be registered?	All write input ports and read output ports
	Create one clock enable signal for each signal	Not selected
	Create an 'aclr' asynchronous clear for the registered ports	Not selected
7	Mixed Port Read-During-Write for Single Input Clock RAM	Old memory contents appear
8	Port A Read-During-Write Option	New Data
	Port B Read-During-Write Option	Old Data

The `top_dpram.v` is a design variation file that contains the top level that instantiates two encoders, a true dual-port RAM, and two decoders. To simulate the design, a testbench, `true_dp_ram.vt`, is created for you to run in the ModelSim®-Altera software.

Functional Simulation in the ModelSim-Altera Software

Simulate the design in the ModelSim-Altera software to generate a waveform that displays the functional behavior of the design example. If you are unfamiliar with the ModelSim-Altera software, refer to the [ModelSim-Altera Software Support](#) page on the Altera website. On the support page, there are links to such topics as installation, usage, and troubleshooting. The following design example provides you with the necessary steps to run the functional simulation even if you are not familiar with the ModelSim-Altera software.

Set up and simulate the design in the ModelSim-Altera software by performing the following steps:

1. Unzip the **Internal_Memory_DesignExample.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files.
5. Click **OK**.
6. On the Tools menu, point to **TCL** and click **Execute Macro**. The **Execute Do File** dialog box appears.
7. Select the **true_dp.do** file and click **Open**. The **true_dp.do** file is a script file that automates all the necessary settings, compiles and simulates the design files, and displays the simulation waveform.
8. Verify the result shown in the **Waveform Viewer** window.

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in **true_dp.do** accordingly.

Understanding the Simulation Results

The top level contains the input and output ports shown in [Table 15](#).

Table 15. Top Level Input and Output Ports Representations (Part 1 of 2)

Ports Name	Ports Type	Descriptions
clock	Input	System Clock for the encoders, RAM, and decoders.
corrupt_dataa_bit0	Input	Registered active high control signal that 'twist' the zero bit (LSB) of input encoded data at port A before writing into the RAM (1)
address_a data_a wren_a rden_a	Input	Address input, data input, write enable, and read enable to port A of the RAM. (1)

Table 15. Top Level Input and Output Ports Representations (Part 2 of 2)

Ports Name	Ports Type	Descriptions
address_b data_b wren_b rden_b	Input	Address input, data input, write enable, and read enable to port B of the RAM. (1)
rdata1 err_corrected1 err_detected1 err_fatal1	Output	Output data read from port A of the RAM, and the ECC-status signals reflecting the data read. (2)
rdata2 err_corrected2 err_detected2 err_fatal2	Output	Output data read from port B of the RAM, and the ECC-status signals reflecting the data read. (2)

Notes to Table 15:

- (1) For input ports, only data signal goes through the encoder; others bypass the encoder and directly go to the RAM block. Since the encoder uses one pipeline, those signal that bypass the encoder require additional pipeline before going to the RAM. This has been implemented in the top level.
- (2) The encoder and decoder each use one pipeline while the RAM uses two pipelines, making the total pipeline equal to four. Therefore, read data is only shown at output ports four clock cycles after the read enable is initiated.

Figure 11 shows the expected simulation waveform results in the ModelSim-Altera software.

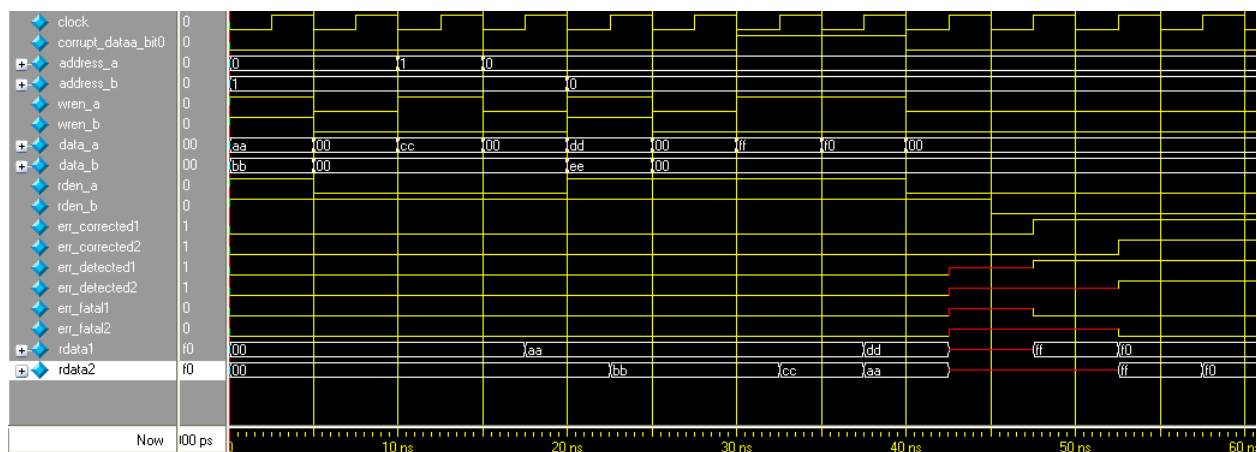
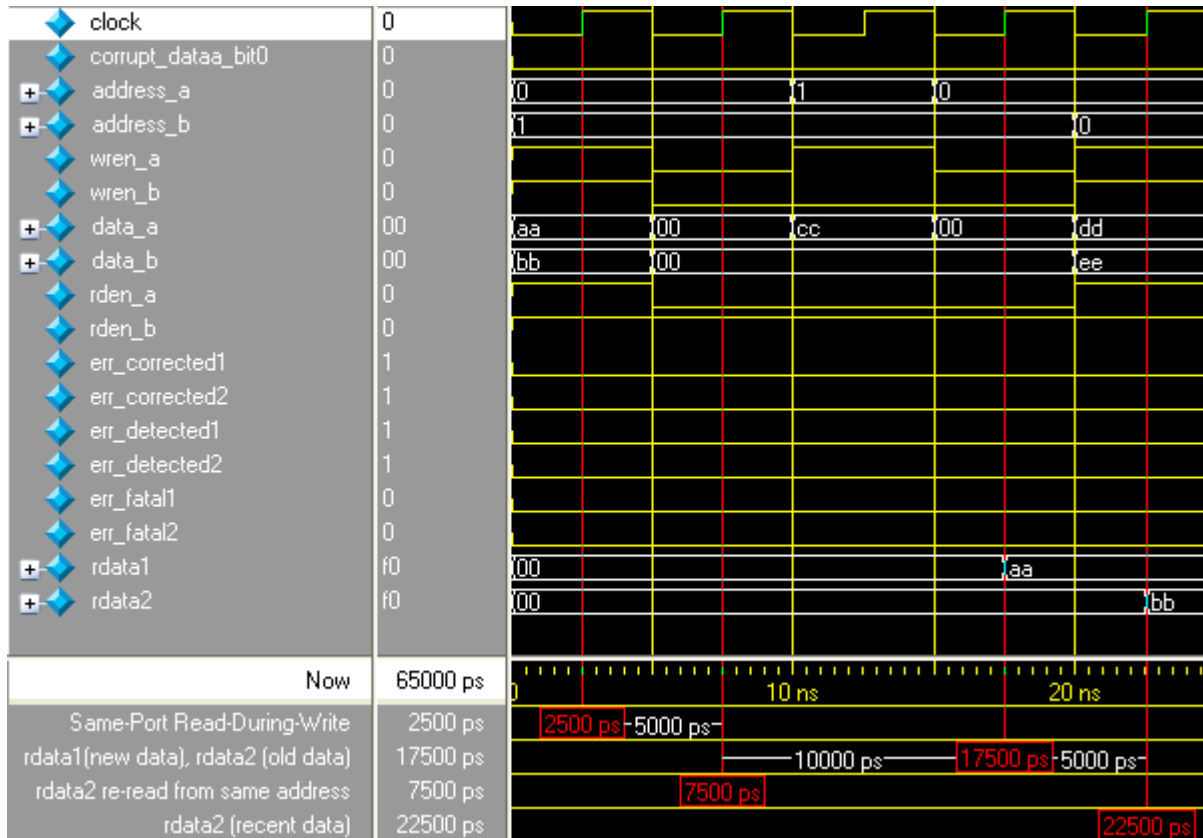
Figure 11. Simulation Results

Figure 12 shows the magnified portion of when the same-port read-during-write occurs for each port A and port B of the RAM.

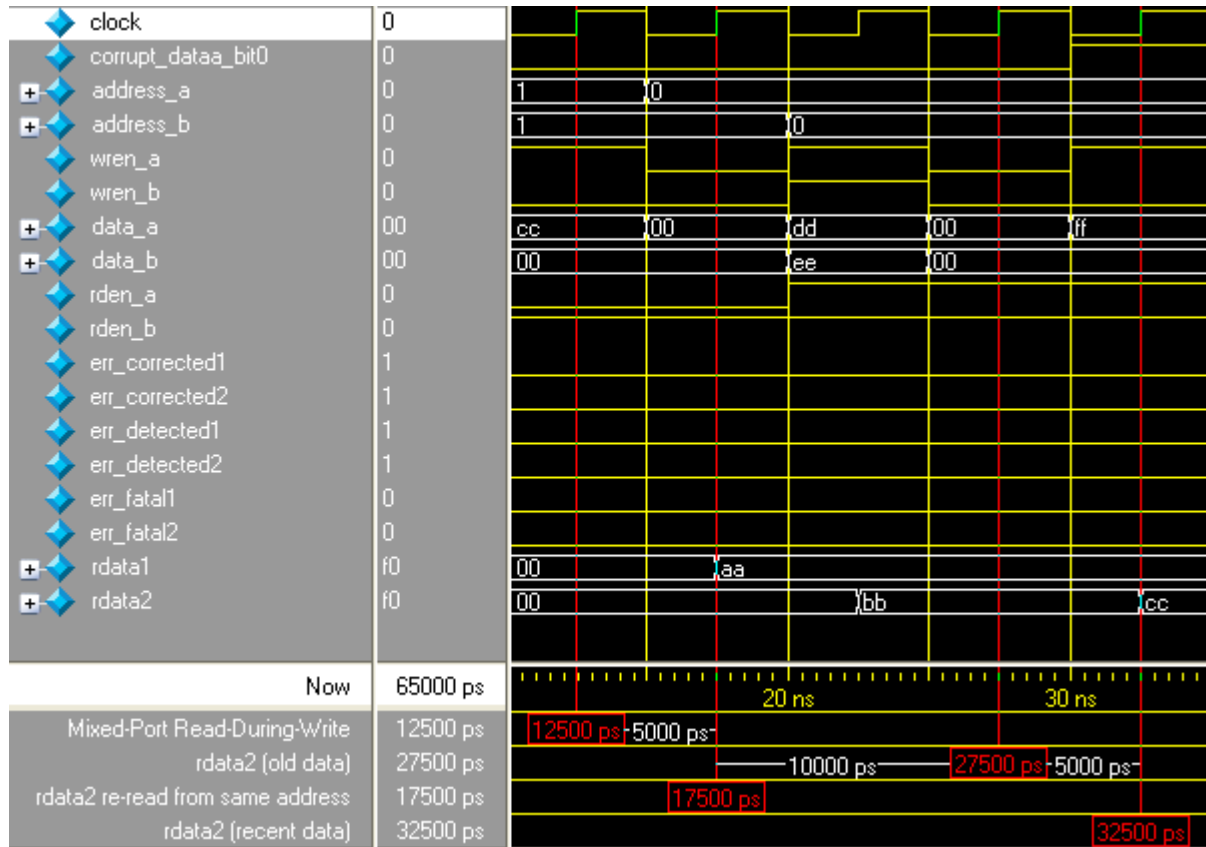
Figure 12. Same-Port Read-During-Write



At 2500 ps, same-port read-during-write occurs for each port A and port B. Since the true dual-port RAM configured to port A is reading the new data and port B is reading the old data when the same-port read-during-write occurs, the rdata1 port shows the new data aa and the rdata2 port shows the old data 00 after four clock cycles at 17500 ps. When the data is read again from the same address at the next rising clock edge at 7500 ps, the rdata2 port shows the recent data bb at 22500 ps.

Figure 13 shows the magnified portion of when the mixed-port read-during-write occurs.

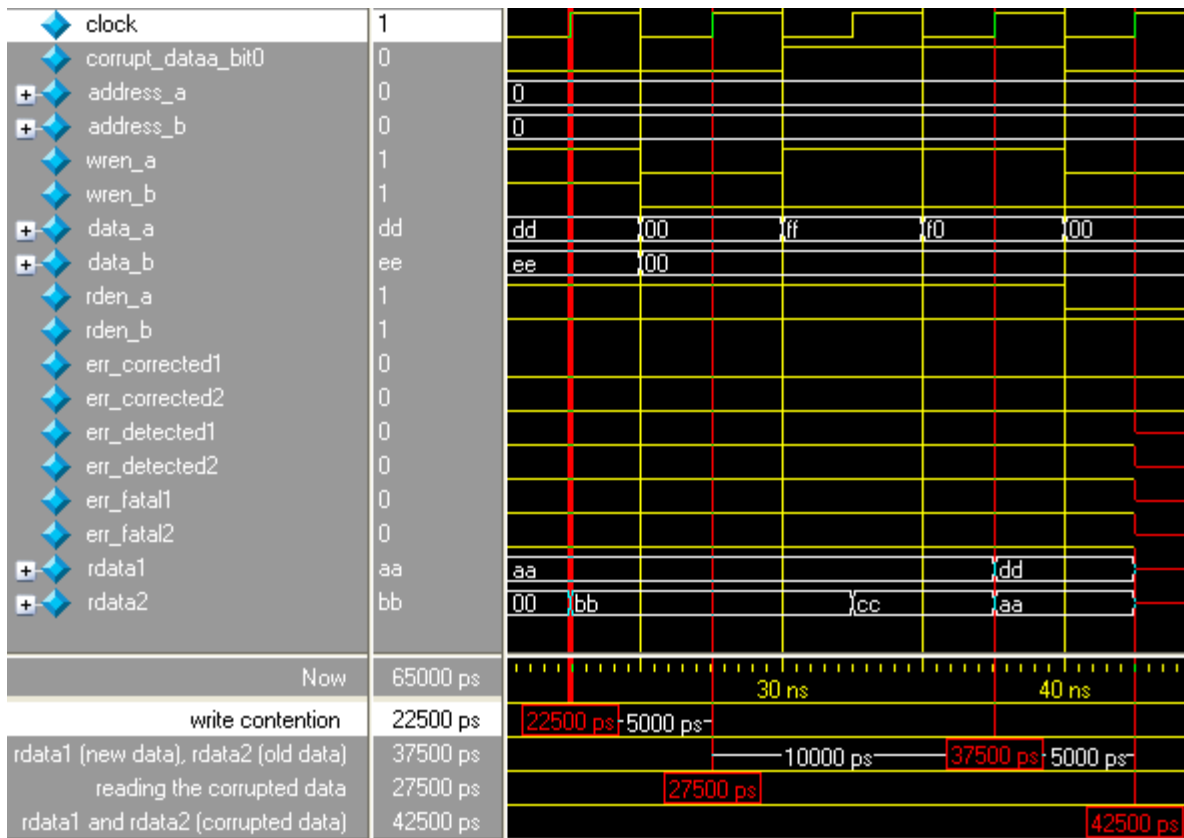
Figure 13. Mixed-Port Read-During-Write



At 12500 ps, mixed-port read-during-write occurs when data cc is both written to port A, and is reading from port B, simultaneously targeting the same address 1. Because the true dual-port RAM that is configured to mixed-port read-during-write is showing the old data, the rdata2 port shows the old data bb after four clock cycles at 27500 ps. When the data is read again from the same address at the next rising clock edge at 17500 ps, the rdata2 port shows the recent data cc at 32500 ps.

Figure 14 shows the magnified portion of when the write contention occurs.

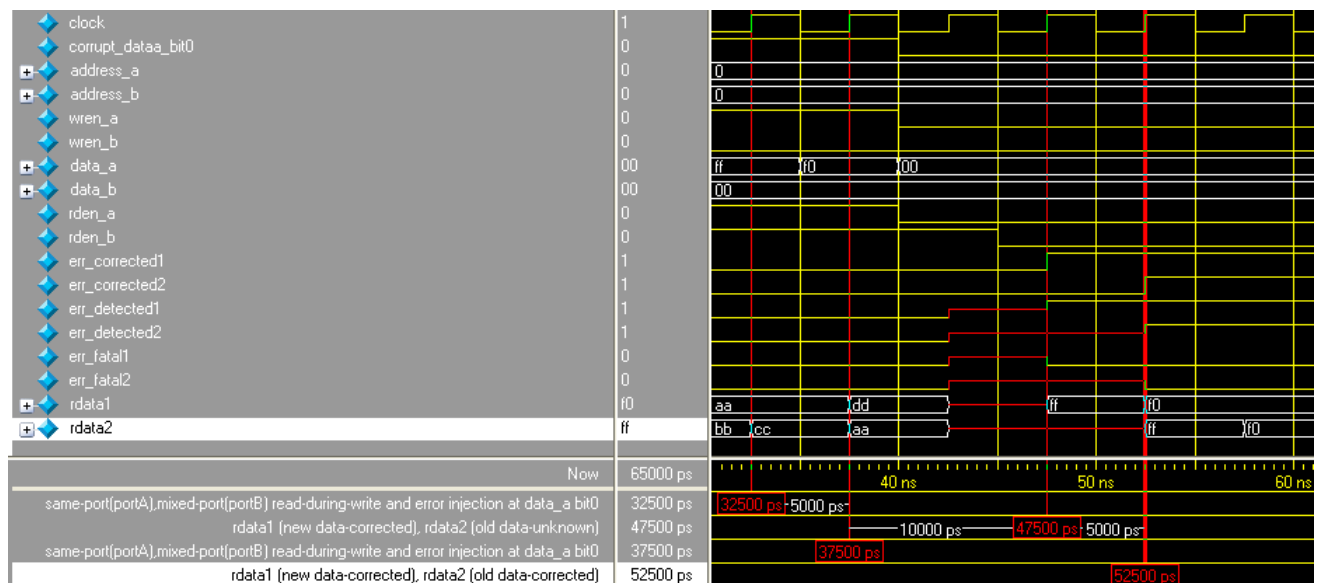
Figure 14. Write Contention




At 22500 ps, the write contention occurs when data dd and ee are written to address 0 simultaneously. Besides that, the same-port read-during-write also occurs for port A and port B. The setting for port A and port B for same-port read-during-write takes effect when the rdata1 port shows the new data dd and the rdata2 port shows the old data aa after four clock cycles at 37500 ps. When the data is read again from the same address at the next rising clock edge at 27500 ps, rdata1 and rdata2 ports show unknown values at 42500 ps. Apart from that, the unknown data input to the decoder also results in an unknown ECC status.

Figure 15 shows the magnified portion of the effect when an error is injected to twist the LSB of the encoded data at port A by asserting `corrupt_dataa_bit0`.

Figure 15. Error Injection– Asserting `corrupt_dataa_bit0`



At 32500 ps, same-port read-during-write occurs at port A while mixed-port read-during-write occurs at port B. The `corrupt_dataa_bit0` is also asserted to corrupt the LSB of encoded data at port A; therefore, the storing data has the LSB corrupted, in which the intended data `ff` is corrupted, becomes `fe`, and stored at address 0. After four clock cycles at 47500 ps, the `rdata1` port shows the new data `ff` that has been corrected by the decoder, and the ECC status signals, `err_corrected1` and `err_detected1`, are asserted. For `rdata2` port, old data (which is unknown) is shown and the ECC-status signal remains unknown.

 The decoders only correct the single-bit error of the data shown at `rdata1` and `rdata2` ports. The actual data stored at address 0 in the RAM remains corrupted, until new data is written.

At 37500 ps, the same condition happens to port A and port B. The difference is port B reads the corrupted old data `fe` from address 0. After four clock cycles at 52500 ps, the `rdata2` port shows the old data `ff` that has been corrected by the decoder and the ECC status signals, `err_corrected2` and `err_detected2`, are asserted to show the data has been corrected.

Ports and Parameters

The ports and parameters details are important if you bypass the MegaWizard Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in your design. The details of the parameters are hidden from the MegaWizard Plug-In Manager interface.

The following list are two commonly used megafunctions if you decide to instantiate the megafunctions:

- ALTSYNCRAM megafunction (Refer to [Table 16 on page 28](#) for the input and output ports and [Table 17 on page 34](#) for the parameters information).
- ALTDPRAM megafunction (Refer to [Table 18 on page 42](#) for the input and output ports and [Table 19 on page 45](#) for the parameters information).

Altera recommends you to use ALTSYNCRAM megafunction to build synchronous memory function for single-port RAM, dual-port RAM, single-port ROM, and dual-port ROM. Use ALTDPRAM megafunction if you want to create a asynchronous read dual-port RAM support.

[Table 16](#) shows the input and output ports for the ALTSYNCRAM megafunction.

Table 16. ALTSYNCRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description
data_a	Input	Optional	Data input to port A of the memory. The data_a port is required if the operation_mode is set to any of the following values: <ul style="list-style-type: none"> ■ SINGLE_PORT ■ DUAL_PORT ■ BIDIR_DUAL_PORT
address_a	Input	Yes	Address input to port A of the memory. The address_a port is required for all operation modes.
wren_a	Input	Optional	Write enable input for address_a port. The wren_a port is required if the operation_mode is set to any of the following values: <ul style="list-style-type: none"> ■ SINGLE_PORT ■ DUAL_PORT ■ BIDIR_DUAL_PORT
rden_a	Input	Optional	Read enable input for address_a port. The rden_a port is supported depending on your selected memory mode and memory block. For more information about the read enable feature, refer to “Read Enable” on page 15 .

Table 16. ALTSYNCRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description
byteena_a	Input	Optional	<p>Byte enable input to mask the data_a port so that only specific bytes, nibbles, or bits of the data are written.</p> <p>The byteena_a port is not supported in the following conditions:</p> <ul style="list-style-type: none"> ■ If implement_in_les parameter is set to ON ■ If operation_mode parameter is set to ROM <p>For more information about byte enable feature and the criterion that you must follow to use the feature correctly, refer to “Byte Enable” on page 13.</p>
addresstall_a	Input	Optional	<p>Address clock enable input to hold the previous address of address_a port for as long as the addresstall_a port is high.</p> <p>The addresstall_a port is only supported in Stratix II, Cyclone II, Arria GX, and newer devices.</p> <p>For more information about address clock enable feature, refer to “Address Clock Enable” on page 12.</p>
q_a	Output	Yes	<p>Data output from port A of the memory.</p> <p>The q_a port is required if the operation_mode parameter is set to any of the following values:</p> <ul style="list-style-type: none"> ■ SINGLE_PORT ■ BIDIR_DUAL_PORT ■ ROM <p>The width of q_a port must be equal to the width of data_a port.</p>
data_b	Input	Optional	<p>Data input to port B of the memory.</p> <p>The data_b port is required if the operation_mode parameter is set to BIDIR_DUAL_PORT.</p>
address_b	Input	Optional	<p>Address input to port B of the memory.</p> <p>The address_b port is required if the operation_mode parameter is set to the following values:</p> <ul style="list-style-type: none"> ■ DUAL_PORT ■ BIDIR_DUAL_PORT
wren_b	Input	Yes	<p>Write enable input for address_b port.</p> <p>The wren_b port is required if operation_mode is set to BIDIR_DUAL_PORT.</p>
rden_b	Input	Optional	<p>Read enable input for address_b port.</p> <p>The rden_b port is supported depending on your selected memory mode and memory block</p> <p>For more information about the read enable feature, refer to “Read Enable” on page 15.</p>

Table 16. ALTSYNCRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description
byteena_b	Input	Optional	<p>Byte enable input to mask the data_b port so that only specific bytes, nibbles, or bits of the data are written.</p> <p>The byteena_b port is not supported in the following conditions:</p> <ul style="list-style-type: none"> ■ If implement_in_les parameter is set to ON ■ If operation_mode parameter is set to SINGLE_PORT, DUAL_PORT, or ROM <p>For more information about byte enable feature and the criterion that you must follow to use the feature correctly, refer to “Byte Enable” on page 13.</p>
addresstall_b	Input	Optional	<p>Address clock enable input to hold the previous address of address_b port for as long as the addresstall_b port is high.</p> <p>The addresstall_b port is only supported in Stratix II, Cyclone II, Arria GX, and newer devices.</p> <p>For more information about address clock enable feature, refer to “Address Clock Enable” on page 12.</p>
q_b	Output	Yes	<p>Data output from port B of the memory.</p> <p>The q_b port is required if the operation_mode is set to the following values:</p> <ul style="list-style-type: none"> ■ DUAL_PORT ■ BIDIR_DUAL_PORT <p>The width of q_b port must be equal to the width of data_b port.</p>

Table 16. ALTSYNCRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description										
clock0	Input	Yes	<p>The following table describes which of your memory clock must be connected to the clock0 port, and port synchronization in different clocking modes:</p> <table border="1"> <thead> <tr> <th>Clocking Mode</th> <th>Descriptions</th> </tr> </thead> <tbody> <tr> <td>Single clock</td> <td>Connect your single source clock to clock0 port. All registered ports are synchronized by the same source clock.</td> </tr> <tr> <td>Read/Write</td> <td>Connect your write clock to clock0 port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock.</td> </tr> <tr> <td>Input Output</td> <td>Connect your input clock to clock0 port. All registered input ports are synchronized by the input clock.</td> </tr> <tr> <td>Independent clock</td> <td>Connect your port A clock to clock0 port. All registered input and output ports of port A are synchronized by the port A clock</td> </tr> </tbody> </table>	Clocking Mode	Descriptions	Single clock	Connect your single source clock to clock0 port. All registered ports are synchronized by the same source clock.	Read/Write	Connect your write clock to clock0 port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock.	Input Output	Connect your input clock to clock0 port. All registered input ports are synchronized by the input clock.	Independent clock	Connect your port A clock to clock0 port. All registered input and output ports of port A are synchronized by the port A clock
Clocking Mode	Descriptions												
Single clock	Connect your single source clock to clock0 port. All registered ports are synchronized by the same source clock.												
Read/Write	Connect your write clock to clock0 port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock.												
Input Output	Connect your input clock to clock0 port. All registered input ports are synchronized by the input clock.												
Independent clock	Connect your port A clock to clock0 port. All registered input and output ports of port A are synchronized by the port A clock												

Table 16. ALTSYNCRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description										
clock1	Input	Optional	<p>The following table describes which of your memory clock must be connected to the clock1 port, and port synchronization in different clocking modes:</p> <table border="1"> <thead> <tr> <th>Clocking Mode</th> <th>Descriptions</th> </tr> </thead> <tbody> <tr> <td>Single clock</td> <td>Not applicable. All registered ports are synchronized by clock0 port.</td> </tr> <tr> <td>Read/Write</td> <td>Connect your read clock to clock1 port. All registered ports related to read operation, such as address_b port, rden_b port, and q_b port are synchronized by the read clock.</td> </tr> <tr> <td>Input Output</td> <td>Connect your output clock to clock1 port. All the registered output ports are synchronized by the output clock.</td> </tr> <tr> <td>Independent clock</td> <td>Connect your port B clock to clock1 port. All registered input and output ports of port B are synchronized by the port B clock.</td> </tr> </tbody> </table>	Clocking Mode	Descriptions	Single clock	Not applicable. All registered ports are synchronized by clock0 port.	Read/Write	Connect your read clock to clock1 port. All registered ports related to read operation, such as address_b port, rden_b port, and q_b port are synchronized by the read clock.	Input Output	Connect your output clock to clock1 port. All the registered output ports are synchronized by the output clock.	Independent clock	Connect your port B clock to clock1 port. All registered input and output ports of port B are synchronized by the port B clock.
Clocking Mode	Descriptions												
Single clock	Not applicable. All registered ports are synchronized by clock0 port.												
Read/Write	Connect your read clock to clock1 port. All registered ports related to read operation, such as address_b port, rden_b port, and q_b port are synchronized by the read clock.												
Input Output	Connect your output clock to clock1 port. All the registered output ports are synchronized by the output clock.												
Independent clock	Connect your port B clock to clock1 port. All registered input and output ports of port B are synchronized by the port B clock.												
clocken0	Input	Optional	Clock enable input for clock0 port.										
clocken1	Input	Optional	Clock enable input for clock1 port.										

Table 16. ALTSYNCRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description
aclr0 aclr1	Input	Optional	<p>Asynchronously clear the registered input and output ports. The <code>aclr0</code> port affects the registered ports that are clocked by <code>clock0</code> clock, while the <code>aclr1</code> port affects the registered ports that are clocked by <code>clock1</code> clock.</p> <p>The asynchronous clear effect on the registered ports can be controlled through their corresponding asynchronous clear parameter, such as <code>outdata_aclr_a</code>, <code>address_aclr_a</code>, and so on.</p> <p>For more information about the asynchronous clear parameters, refer to Table 17.</p>
eccstatus	Output	Optional	<p>A 3-bit wide error correction status port.</p> <p>Indicate whether the data that is read from the memory has an error in single-bit with correction, fatal error with no correction, or no error bit occurs.</p> <p>The <code>eccstatus</code> port is supported if all the following conditions are met:</p> <ul style="list-style-type: none"> ■ <code>operation_mode</code> parameter is set to <code>DUAL_PORT</code> ■ <code>ram_block_type</code> parameter is set to <code>M144K</code> ■ <code>width_a</code> and <code>width_b</code> parameter have the same value ■ Byte enable is not used <p>For more information about the ECC features, restrictions, and the output status definitions, refer to “Error Correction Code (ECC)” on page 18.</p>

Table 17 shows the parameters for the ALTSYNCRAM megafunction.

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
operation_mode	String	Yes	<p>A required parameter to specify the operation of the memory depending on the following memory modes that you want to create:</p> <ul style="list-style-type: none"> ■ SINGLE_PORT for single-port RAM ■ DUAL_PORT for simple dual-port RAM ■ BIDIR_DUAL_PORT for true dual-port RAM or dual-port ROM ■ ROM for single-port ROM <p>For more information about various memory modes, refer to “Memory Modes” on page 2.</p>
width_a width_b	Integer	Optional	<p>Optional parameters (depending on the port usage) to specify the width of the following data input port and data output port:</p> <ul style="list-style-type: none"> ■ width_a for data_a and q_a ports ■ width_b for data_b and q_b ports <p>Mixed-width port is supported when the value for width_a and width_b parameter is different. The mixed-width port is only supported when operation_mode is set to the following values:</p> <ul style="list-style-type: none"> ■ DUAL_PORT ■ BIDIR_DUAL_PORT <p>excluding the following conditions:</p> <ul style="list-style-type: none"> ■ the implement_in_les parameter is set to ON ■ the ram_block_type parameter is set to MLAB <p>For more information about mixed-width port feature, refer to “Mixed-width Port” on page 9.</p>
widthad_a widthad_b	Integer	Optional	<p>Optional parameters (depending on the port usage) to specify the width of the following address input ports:</p> <ul style="list-style-type: none"> ■ widthad_a for address_a port ■ widthad_b for address_b port <p>The parameters are related to numwords_a and numwords_b parameters as follows:</p> <ul style="list-style-type: none"> ■ widthad_a = \log_2 numwords_a ■ widthad_b = \log_2 numwords_b

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
numwords_a numwords_b	Integer	Optional	<p>Optional parameters (depending on the port usage) to specify the memory depth according to the following address ports:</p> <ul style="list-style-type: none"> ■ numwords_a as memory depth for port A. This parameter also represents the actual memory depth ■ numwords_b as memory depth for port B <p>For same-width port, the parameters must have the same value.</p> <p>For mixed-width port, numwords_b parameter has a different value from the numwords_a parameter, and is derived as follows:</p> <ul style="list-style-type: none"> ■ If width_b > width_a, then numwords_b = numwords_a / (width_b / width_a) ■ If width_a > width_b, then numwords_b = numwords_a x (width_a / width_b) <p>The parameter that represents the memory depth can be in non-power of two but the actual memory depth might be different. For more information about the effects on having the memory depth in non-power of two, refer to “Port Width Configuration (Memory Depth x Data Width)” on page 8.</p>
byte_size	Integer	Optional	<p>Optional parameter (depending on the byte enable usage) to specify the byte size for the byte-enable mode. The supported values are 5, 8, 9 and 10. Values 5 and 10 are only supported if ram_block_type is set to MLAB.</p> <p>For more information about the byte size, refer to “Byte Enable” on page 13.</p>
width_byteena_a	Integer	Optional	<p>Optional parameter (depending on the byte enable usage on data_a port) to specify the width of the byteena_a port.</p> <p>The width_byteena_a parameter value must be equal to width_a / byte_size.</p>
width_byteena_b	Integer	Optional	<p>Optional parameter (depending on byte enable usage on data_b port) to specify the width of the byteena_b port.</p> <p>The width_byteena_b parameter value must be equal to width_b / byte_size.</p>

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
ram_block_type	String	Optional	<p>Optional parameter to specify the TriMatrix Memory Block type to use depending on your target device. The following list are the available values:</p> <ul style="list-style-type: none"> ■ M512 ■ M4K ■ M-RAM ■ MLAB ■ M9K ■ M144K ■ AUTO <p>By choosing <code>AUTO</code>, the compiler has the flexibility to decide on the suitable memory block that results in optimum resource usage and best performance.</p> <p>For more information on the availability of the memory block in different memory modes, and the benefit of choosing <code>AUTO</code>, refer to “Memory Block Types” on page 5.</p>
read_during_write_mode_mixed_ports	String	Optional	<p>Optional parameter to specify the behavior when the read and write operations occur at different ports on the same RAM address. The values are:</p> <ul style="list-style-type: none"> ■ <code>OLD_DATA</code> ■ <code>NEW_DATA</code> (only supported by MLAB) ■ <code>DONT_CARE</code> <p>The parameter and values are applicable depending on the memory mode and target memory block that you use. For more information about when mixed-port read-during-write behavior is supported and the applicable values, refer to “Read-During-Write” on page 16.</p>

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
read_during_write_mode_port_a read_during_write_mode_port_b	String	Optional	<p>Use the <code>read_during_write_mode_port_a</code> parameter to specify the output behavior of port A when the read and write operations occur at the same port A and on the same RAM address. The same way is applied to the output of port B using the <code>read_during_write_mode_port_b</code> parameter.</p> <p>The valid values differs for different memory modes and target memory block used. In general, the values are:</p> <ul style="list-style-type: none"> ■ <code>OLD_DATA</code> ■ <code>DONT_CARE</code> ■ <code>NEW_DATA_NO_NBE_READ</code> (the output port shows x for write masked bytes instead of old data when byte enable is used) ■ <code>NEW_DATA_WITH_NBE_READ</code> (the output port shows old data for write masked bytes when byte enable is used) <p>For more information about same-port read-during-write behavior and the values applicable for a particular memory mode and memory block, refer to “Read-During-Write” on page 16.</p>

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
indata_aclr_a indata_aclr_b address_aclr_a address_aclr_b byteena_aclr_a byteena_aclr_b wrcontrol_aclr_a wrcontrol_aclr_b	String	Optional	<p>Optional parameters to specify which input registered ports are affected by asynchronous clear. Values are <code>CLEAR0</code> and <code>NONE</code>. The following list shows the respective parameters and controlled ports:</p> <ul style="list-style-type: none"> ■ <code>indata_aclr_a</code>—<code>data_a</code> port ■ <code>indata_aclr_b</code>—<code>data_b</code> port ■ <code>address_aclr_a</code>—<code>address_a</code> port ■ <code>address_aclr_b</code>—<code>address_b</code> port ■ <code>byteena_aclr_a</code>—<code>byteena_a</code> port ■ <code>byteena_aclr_b</code>—<code>byteena_b</code> port ■ <code>wrcontrol_aclr_a</code>—<code>wren_a</code> port ■ <code>wrcontrol_aclr_b</code>—<code>wren_b</code> port <p>The asynchronous clear might not affect all the input registered ports and thus some of the parameters might not be applicable.</p> <p>For Stratix II, Stratix II GX, Cyclone II, and Arria GX devices, all the parameters are not applicable.</p> <p>For Stratix III, Cyclone III, Arria II GX, and newer devices, the <code>CLEAR0</code> is only applicable for:</p> <ul style="list-style-type: none"> ■ <code>address_aclr_b</code> parameter if <code>operation_mode</code> is set to <code>DUAL_PORT</code> ■ <code>address_aclr_a</code> parameter if <code>operation_mode</code> is set to <code>ROM</code> <p>Set the parameters to <code>NONE</code> whenever the parameters are not applicable.</p> <p>For more information about registered input ports that can be affected by the asynchronous clear port for different target devices, memory mode, and memory block, refer to “Asynchronous Clear” on page 14.</p>
outdata_aclr_a outdata_aclr_b eccstatus_aclr	String	Optional	<p>Optional parameters to specify which output registered ports are affected by asynchronous clear. The values are <code>CLEAR0</code>, <code>CLEAR1</code>, and <code>NONE</code>. The <code>clear0</code> affects the registers clocked by <code>clock0</code> while <code>clear1</code> affects the registers clocked by <code>clock1</code>. The following list shows the respective parameters and controlled ports:</p> <ul style="list-style-type: none"> ■ <code>outdata_aclr_a</code>—<code>q_a</code> port ■ <code>outdata_aclr_b</code>—<code>q_b</code> port ■ <code>eccstatus_aclr</code>—<code>eccstatus</code> port

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
indata_reg_b address_reg_b byteena_reg_b wrcontrol_wraddress_reg_b	String	Optional	Optional parameter to specify the clock used by port B. The values are <code>CLOCK0</code> and <code>CLOCK1</code> . The following list shows the respective parameters and controlled ports: <ul style="list-style-type: none"> ■ <code>indata_reg_b</code>—<code>data_b</code> port ■ <code>address_reg_b</code>—<code>address_b</code> port (<code>address_reg_b</code> serves as the reference for the clock source of port B if other parameters are not specified or are different from <code>address_reg_b</code>.) ■ <code>byteena_reg_b</code>—<code>byteena_b</code> port ■ <code>wrcontrol_wraddress_reg_b</code>—<code>wren_b</code> port
outdata_reg_a outdata_reg_b eccstatus_reg	String	Optional	Optional parameters to specify the clock used for the output ports. The values are <code>CLOCK0</code> , <code>CLOCK1</code> , and <code>UNREGISTERED</code> . The following list shows the respective parameters and controlled ports: <ul style="list-style-type: none"> ■ <code>outdata_reg_a</code>—<code>q_a</code> port ■ <code>outdata_reg_b</code>—<code>q_b</code> port ■ <code>eccstatus_reg</code>—<code>eccstatus</code> port
init_file	String	Optional	Optional parameter to specify the name of the Memory Initialization File (<code>.mif</code>) or Hexadecimal (Intel-Format) Output File (<code>.hex</code>) containing memory initialization data (<code><file name></code>). The default is <code>UNUSED</code> . If omitted, the default value for all contents is <code>0</code> . If you want to initialize the content to <code>x</code> on power-up simulation, refer to <code>power_up_uninitialized</code> parameter. The following list the restrictions for this parameter: <ul style="list-style-type: none"> ■ The <code>init_file</code> parameter is no applicable when the <code>ram_block_type</code> parameter is set to <code>M-RAM</code> ■ The <code>init_file</code> parameter is no applicable for HardCopy series of devices ■ When the <code>operation_mode</code> parameter is set to <code>DUAL_PORT</code>, the Compiler uses only the <code>width_b</code> parameters to read the initialization file ■ The <code>init_file</code> parameter is required if you want to create single-port or dual-port ROM

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
<code>init_file_layout</code>	String	Optional	<p>Optional parameter to specify the layout port that is used with the initialization file. The values are:</p> <ul style="list-style-type: none"> ■ <code>PORT_A</code> ■ <code>PORT_B</code> ■ <code>UNUSED</code> <p>The initialized memory content will have the same width as the width of the port assigned.</p> <p>If omitted, the default value is <code>UNUSED</code>. If the <code>operation_mode</code> is set to <code>DUAL_PORT</code> mode, the default value is <code>PORT_B</code>. If the <code>operation_mode</code> is set to other modes, the default value is <code>PORT_A</code>.</p>
<code>maximum_depth</code>	Integer	Optional	<p>Optional parameter to specify the slicing depth of the RAM slices. The values must be in power of two, and the range of the values depends on the target memory block.</p> <p>For more information about the supported value range for different target memory block, and the usage of slicing the memory block, refer to “Maximum Block Depth” on page 9.</p>
<code>intended_device_family</code>	String	Optional	Optional parameter to specify the target device family. This parameter is used for modeling and behavioral simulation purposes.
<code>lpm_type</code>	String	Optional	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.
<code>clock_enable_input_a</code> <code>clock_enable_input_b</code> <code>clock_enable_output_a</code> <code>clock_enable_output_b</code> <code>clock_enable_eccstatus</code>	String	Optional	<p>Optional parameters to specify the clock enable for the ports. The values are:</p> <ul style="list-style-type: none"> ■ <code>NORMAL</code> (clock enable is used) ■ <code>BYPASS</code> (clock enable is not used) <p>The following list shows the respective parameters and controlled ports:</p> <ul style="list-style-type: none"> ■ <code>clock_enable_input_a</code>—input registers of A port. ■ <code>clock_enable_input_b</code>—input registers of B port. ■ <code>clock_enable_output_a</code>—output registers of A port. ■ <code>clock_enable_output_b</code>—output registers of B port ■ <code>clock_enable_eccstatus</code>—output registers of <code>eccstatus</code> port

Table 17. ALTSYNCRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
<code>enable_ecc</code>	String	Optional	<p>Optional parameter to specify whether the ECC feature is on or off.</p> <p>The values are:</p> <ul style="list-style-type: none"> ■ TRUE ■ FALSE <p>For more information about the ECC features, restrictions, and the output status definitions, refer to “Error Correction Code (ECC)” on page 18.</p>
<code>power_up_uninitialized</code>	String	Optional	<p>Optional parameter to specify whether to initialize memory content data to X on power-up simulation.</p> <p>The value are:</p> <ul style="list-style-type: none"> ■ TRUE ■ FALSE <p>For M-RAM, only the TRUE value is valid. The <code>init_file</code> parameter has higher priority than the <code>power_up_uninitialized</code> parameter.</p>
<code>implement_in_les</code>	String	Optional	<p>Optional parameter to specify the usage of RAM blocks. The <code>implement_in_les</code> parameter is only applicable if the <code>operation_mode</code> is set to any of the following values:</p> <ul style="list-style-type: none"> ■ SINGLE_PORT ■ DUAL_PORT <p>The valid values for the <code>implement_in_les</code> parameter are:</p> <ul style="list-style-type: none"> ■ ON ■ OFF <p>Set the value to OFF will implement your RAM using logic cells that has write triggering at the falling clock edge as M512 memory block. The logic cells implementation also affects the memory performance. If embedded memory block is sufficient, you must ignore the <code>implement_in_les</code> parameter, and set the <code>ram_block_type</code> parameter to AUTO.</p> <p>For more information about the embedded memory block and their advantages over logic cells implementation, refer to “Memory Block Types” on page 5.</p>

Altera recommends you to use ALTDPRAM megafunction when you want to create simple dual-port RAM (true dual-port RAM does not support the following conditions) in any of the following conditions:

- when you implement memory using MLAB with un-registered rdaddress port (only MLAB supports asynchronous read operation)
- when you implement memory using logic element with no M512 emulation



Logic element with M512 emulation behave like M512 memory block, in which write operation triggers at falling clock edge.

Use the ALTSYNCRAM megafunction if your memory specification does not meet any of the conditions mentioned in the previous list.



The ports and parameters description for the ALTDPRAM megafunction in the following sections only include information that is applicable to any of the mentioned conditions that are also recommended for the ALTDPRAM megafunction.

Table 18 shows the input and output ports for the ALTDPRAM megafunction.

Table 18. ALTDPRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description
data	Input	Yes	Data input to the memory. The data port is required and the width must be equal to the width of the q port.
wraddress	Input	Yes	Write address input to the memory. The wraddress port is required and must be equal to the width of the raddress port.
wren	Input	Yes	Write enable input for wraddress port. The wren port is required.
raddress	Input	Yes	Read address input to the memory. The rdaddress port is required and must be equal to the width of wraddress port.
rden	Input	Optional	Read enable input for raddress port. The rden port is supported when the use_eab parameter is set to OFF. The rden port is not supported when the ram_block_type parameter is set to MLAB. Instantiate the ALTSYNCRAM megafunction if you want to use read enable feature with other memory blocks.
byteena	Input	Optional	Byte enable input to mask the data port so that only specific bytes, nibbles, or bits of data are written. The byteena port is not supported when use_eab parameter is set to OFF. It is supported in Arria II GX, Stratix III, Cyclone III, and newer devices with the ram_block_type parameter set to MLAB. For more information about byte enable feature and the criterion that you must follow to use the feature correctly, refer to “Byte Enable” on page 13.

Table 18. ALTDPRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description								
wraddressstall	Input	Optional	Write address clock enable input to hold the previous write address of wraddress port for as long as the wraddressstall port is high. The wraddressstall port is only supported in Stratix II, Cyclone II, Arria GX, and newer devices. For more information about address clock enable feature, refer to “Address Clock Enable” on page 12 .								
rdaddressstall	Input	Optional	Read address clock enable input to hold the previous read address of rdaddress port for as long as the wraddressstall port is high. The rdaddressstall port is only supported in Stratix II, Cyclone II, Arria GX, and newer devices except when the rdaddress_reg parameter is set to UNREGISTERED. For more information about address clock enable feature, refer to “Address Clock Enable” on page 12 .								
q	Output	Yes	Data output from the memory. The q port is required, and must be equal to the width data port.								
inclock	Input	Yes	The following table describes which of your memory clock must be connected to the inclock port, and port synchronization in different clocking modes: <table border="1" data-bbox="812 1039 1339 1638"> <thead> <tr> <th>Clocking Mode</th> <th>Descriptions</th> </tr> </thead> <tbody> <tr> <td>Single clock</td> <td>Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock.</td> </tr> <tr> <td>Read/Write</td> <td>Connect your write clock to inclock port. All registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port are synchronized by the write clock.</td> </tr> <tr> <td>Input/Output</td> <td>Connect your input clock to inclock port. All registered input ports are synchronized by the input clock.</td> </tr> </tbody> </table>	Clocking Mode	Descriptions	Single clock	Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock.	Read/Write	Connect your write clock to inclock port. All registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port are synchronized by the write clock.	Input/Output	Connect your input clock to inclock port. All registered input ports are synchronized by the input clock.
Clocking Mode	Descriptions										
Single clock	Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock.										
Read/Write	Connect your write clock to inclock port. All registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port are synchronized by the write clock.										
Input/Output	Connect your input clock to inclock port. All registered input ports are synchronized by the input clock.										

Table 18. ALTDPRAM Megafunction Input and Output Ports Description

Port Name	Type	Required	Description								
outclock	Input	Yes	<p>The following table describes which of your memory clock must be connected to the outclock port, and port synchronization in different clocking modes:</p> <table border="1"> <thead> <tr> <th>Clocking Mode</th> <th>Descriptions</th> </tr> </thead> <tbody> <tr> <td>Single clock</td> <td>Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock.</td> </tr> <tr> <td>Read/Write</td> <td>Connect your read clock to outclock port. All registered ports related to read operation, such as rdaddress port, rdren port, and q port are synchronized by the read clock.</td> </tr> <tr> <td>Input/Output</td> <td>Connect your output clock to outclock port. The registered q port is synchronized by the output clock.</td> </tr> </tbody> </table>	Clocking Mode	Descriptions	Single clock	Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock.	Read/Write	Connect your read clock to outclock port. All registered ports related to read operation, such as rdaddress port, rdren port, and q port are synchronized by the read clock.	Input/Output	Connect your output clock to outclock port. The registered q port is synchronized by the output clock.
Clocking Mode	Descriptions										
Single clock	Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock.										
Read/Write	Connect your read clock to outclock port. All registered ports related to read operation, such as rdaddress port, rdren port, and q port are synchronized by the read clock.										
Input/Output	Connect your output clock to outclock port. The registered q port is synchronized by the output clock.										
inclocken	Input	Optional	Clock enable input for inclock port.								
outclocken	Input	Optional	Clock enable input for outclock port.								
aclr	Input	Optional	<p>Asynchronously clear the registered input and output ports. The asynchronous clear effect on the registered ports can be controlled through their corresponding asynchronous clear parameter, such as indata_aclr, wraddress_aclr, and so on.</p> <p>For more information about the asynchronous clear parameters, refer to Table 19.</p>								

Table 19 shows the parameters for the ALTDPRAM megafunction.

Table 19. ALTDPRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
width	Integer	Yes	A required parameter to specify the widths of data port and q port.
widthhad	Integer	Yes	A required parameter to specify the widths of rdaddress port and wraddress port.
numwords	Integer	Yes	A required parameter to specify the memory depth. The value must be within the range $2^{\text{widthhad}-1} < \text{numwords} \leq 2^{\text{widthhad}}$, and must be more than 1. For more information about the effect on having the memory depth in non-power of two, refer to “Port Width Configuration (Memory Depth × Data Width)” on page 8.
byte_size	Integer	Optional	An optional parameter (depending on the byte enable usage) to specify the byte size for the byte-enable mode. The values are 5, 8, 9, and 10. This parameter is not applicable when use_eab parameter is set to OFF. For more information, refer to “Byte Enable” on page 13.
width_byteena	Integer	Optional	An optional parameter (depending on the byte enable usage) to specify the width of the byteena port. This parameter has the following conditions: <ul style="list-style-type: none"> ■ Must be equal to width / byte_size ■ Not applicable when use_eab parameter is set to OFF.
ram_block_type	String	Optional	An optional parameter to specify the TriMatrix Memory Block type to be used. Set the following parameters if you want to create an asynchronous read RAM: <ul style="list-style-type: none"> ■ ram_block_type = MLAB ■ rdaddress_reg = UNREGISTERED ■ rdcontrol_reg = UNREGISTERED ■ outdata_reg = UNREGISTERED Use the ALTSYNCRAM megafunction if you want to use other memory blocks, or if you want to create a synchronous RAM.
read_during_write_mode_mixed_ports	String	Optional	An optional parameter to specify the output behavior of the read/write mode when the read and write operations occur at different ports on the same RAM address. This parameter is only applicable when the use_eab parameter is set to OFF.

Table 19. ALTDPRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
indata_aclr wraddress_aclr rdaddress_aclr wrcontrol_aclr rdcontrol_aclr	String	Optional	<p>An optional parameter to specify which input registered ports are affected by <code>aclr</code> port. Values are <code>ON</code> and <code>OFF</code>. The following list shows the respective parameters and controlled ports:</p> <ul style="list-style-type: none"> ■ <code>indata_aclr</code>—data port ■ <code>wraddress_aclr</code>—wraddress port ■ <code>wrcontrol_aclr</code>—wren port ■ <code>rdaddress_aclr</code>—rdaddress port ■ <code>rdcontrol_aclr</code>—rden port <p>The parameters are only applicable when <code>use_eab</code> parameter is set to <code>OFF</code> and depends on the target devices:</p> <ul style="list-style-type: none"> ■ For Stratix, Stratix GX, and Cyclone devices, all parameters are applicable. ■ For Arria GX, Stratix II, Stratix II GX and Cyclone II, all parameters are not applicable ■ For Arria II GX, Stratix III, Cyclone III, and newer devices, only <code>rdaddress_aclr</code> parameter is applicable <p>Set the parameter to <code>OFF</code> if it is not applicable.</p>
outdata_aclr	String	Optional	<p>An optional parameter to specify whether the <code>aclr</code> port affects the registered <code>q</code> port. Values are <code>ON</code> and <code>OFF</code>.</p>
indata_reg wraddress_reg wrcontrol_reg	String	Yes	<p>A required parameter to specify the clock used by the write ports. Values are <code>UNREGISTERED</code> and <code>INCLOCK</code>. The following list shows the respective parameters and controlled ports:</p> <p><code>indata_reg</code>—data port <code>wraddress_reg</code>—wraddress port <code>wrcontrol_reg</code>—wren port</p> <p>For Stratix, Stratix GX, Cyclone, and newer devices, the legal value is only <code>INCLOCK</code>.</p>
rdaddress_reg rdcontrol_reg	String	Optional	<p>An optional parameter to specify the clock used by the read ports. Values are <code>UNREGISTERED</code>, <code>OUTCLOCK</code>, and <code>INCLOCK</code>. The following list shows the respective parameters and controlled ports:</p> <p><code>rdaddress_reg</code>—rdaddress port <code>rdcontrol_reg</code>—rden port</p> <p>Set the parameters to <code>UNREGISTERED</code> and set the <code>ram_block_type</code> parameter to <code>MLAB</code> if you want to create an asynchronous read RAM.</p> <p>If the <code>use_eab</code> parameter is set to <code>OFF</code>, set the parameter based on whether the <code>rdaddress</code> and <code>rden</code> ports are clocked by <code>inclock</code> port or <code>outclock</code> port.</p>
outdata_reg	String	Yes	<p>A required parameter to specify the clock used by the <code>q</code> port. Values are <code>UNREGISTERED</code>, <code>OUTCLOCK</code>, and <code>INCLOCK</code></p>

Table 19. ALTDPRAM Megafunction Parameters Description

Parameter Name	Type	Required	Description
maximum_depth	Integer	Optional	<p>An optional parameter to specify the slicing depth of the RAM slices.</p> <p>The valid values are 32 and 64 when <code>ram_block_type</code> parameter is set to <code>MLAB</code>. The default value is 0. This parameter is not applicable in the following conditions:</p> <ul style="list-style-type: none"> ■ For Stratix III MLAB ■ If <code>use_eab</code> parameter is set to <code>OFF</code>. <p>For more information about the usage of slicing the memory block, refer to “Maximum Block Depth” on page 9.</p>
lpm_file	String	Optional	<p>An optional parameter to specify the name of the Memory Initialization File (<code>.mif</code>) or Hexadecimal (Intel-Format) Output File (<code>.hex</code>) containing RAM initialization data ("<code><file name></code>"). The default value is <code>UNUSED</code>.</p> <p>If omitted, the default value for all contents is 0.</p> <p>The <code>wren</code> port must be registered to support memory initialization.</p>
intended_device_family	String	Optional	<p>An optional parameter to specify the target device family. This parameter is used for modeling and behavioral simulation purposes.</p>
use_eab	String	Optional	<p>An optional parameter to specify whether to implement memory using TriMatrix Memory Block or logic cells.</p> <p>Legal values are <code>ON</code> and <code>OFF</code>.</p> <p>Setting the <code>use_eab</code> parameter to <code>OFF</code> will result in memory implemented in logic cells and this affects the performance.</p> <p>Logic cells that are implemented have the write triggering at rising clock edge. If you intend to implement logic cells that have write triggering at falling clock edge as M512 memory block, use the <code>ALTSYNCRAM</code> megafunction.</p> <p>The <code>use_eab</code> parameter has higher priority than the <code>ram_block_type</code> parameter.</p>

Revision History

Table 20 table shows the revision history for this user guide.

Table 20.

Date	Version	Changes Made
November 2009	1.0	Initial release



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001